# pygtlink

*Release 1.0.0*

**Mar 17, 2020**

# Contents

CHAPTER 1

# Introduction to pygtlink library!

An introduction to pygtlink library

# CHAPTER 2

# Package Documentation

**class** pygtlink.**IgtlHeader**
Bases: object

**pack**(*endian='>'*)

**unpack**(*binary_header*, *endian='>'*)

**class** pygtlink.**MessageBase**
Bases: object

Implementation of the base message. Base class for all king of messages

**Variables**

- **header** (*bytearray*) – A pointer to the byte array for the serialized header. To prevent large copy of the byte array in the Pack() function, header byte array is concatenated to the byte array for the body.
- **body** (*bytearray*) – A pointer to the byte array for the serialized body. To prevent large copy of the byte array in the Pack() function, header byte array is concatenated to the byte array for the header.
- **_messageSize** (*int*) – The size of the message
- **_bodySize** (*int*) – The size of the body to be read. This function must be called after the message header is set.
- **_messageType** (*str*) – The message type
- **_headerVersion** (*int*) – An unsigned short for the message format version
- **_deviceName** (*str*) – A character string for the device name (message name).
- **_timeStampSec** (*uint64*) – A time stamp (second) for message creation. It consists of fields for seconds (m_TimeStampSec)and fractions of a second (m_TimeStampSecFraction).

- **_timeStampFraction** (*uint64*) – A time stamp (second) for message creation. It consists of fields for seconds (m_TimeStampSec)and fractions of a second (m_TimeStampSecFraction).
- **_isHeaderUnpacked** (*bool*) – Unpacking (deserialization) status for the header
- **_isBodyUnpacked** (*bool*) – Unpacking (deserialization) status for the body
- **_isBodyPacked** (*bool*) – Packing (deserialization) status for the body

**copyHeader**(*messageBase*)
Copies the unpacked header from another message

**Parameters** **messageBase** (pygtlink.MessageBase) – message from which to copy the header

**getBodyCrc**()
Gets the message body crc

**Returns** The body crc

**getBodySizeToRead**()

**Gets the size of the body to be read. This function must be called after the message header is set. Same as** getPackBodySize()

**Returns** The body size to read

**getDeviceName**()
Gets the device name

**Returns** The message device name

**static getHeaderSize**()
Gets the igtl message header size (same for all messages)

**Returns** The igtl message header size

**getHeaderVersion**()
Gets the message version number

**Returns** The message header version

**getMessageType**()
Gets the device type

**Returns** The message type

**getPackBodySize**()
Gets the size of the message body.

**Returns** The body size

**getPackSize**()
Gets the size of the serialized message

**Returns** The message size

**getTimeStamp**()
Gets the message timestamp

**Returns** The timestamp in seconds since the epoch as a floating point number. The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).

**getTimeStampSecFrac**()
Gets the message timestamp

**Returns** The timestamp expressed in second and fraction

**initBuffer**()
Initializes the message

**isHeaderUnpacked**()

**pack**()
Serializes the header and body based on the member variables. PackContent() must be implemented in the child class.

**Returns** 0 in case of error, 1 if the message was successfully packed

**setDeviceName**(*device_name*)
Sets the device name

**Parameters** **device_name** (*str*) – The device name to set

**setHeaderVersion**(*header_version*)
Sets the message version number

**Parameters** **header_version** (*int*) – The header version to set

**setMessageType**(*msg_type*)
Sets the device (message) type

**Parameters** **msg_type** (*str*) – The message type to set

**setTimeStamp**(*timestamp*)
Sets the message timestamp

**Parameters** **timestamp** (*int*) – timestamp in seconds since the epoch as a floating point number. The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0). Can be obtained using the python function time.time() from the time module

**unpack**(*crccheck=0*)
Unpack() deserializes the header and/or body, extracting data from the byte stream. If the header has already been deserialized, Unpack() deserializes only the body part. UnpackBody() must be implemented to deserialize the body part. Unpack() performs 64-bit CRC check, when crccheck = 1.

**Parameters** **crccheck** (*int*) – The body crccheck

**Returns**

The unpacking result, i.e.

| Code | Meaning |
|---|---|
| UNPACK_UNDEF | Nothing deserialized |
| UNPACK_HEADER | The header has been deserialized. |
| UNPACK_BODY | The body has been deserialized. |

If CRC check fails, Unpack() doesn't deserialize the body, thus it doesn't return UNPACK_BODY flag.

**class** pygtlink.**ImageMessage2**
Bases: pygtlink.igtl_message_base.MessageBase

The class implements the openIgtLink image message

**Variables**

- **_dimensions** (*float*) – A vector containing the numbers of voxels in i, j and k directions.
- **_spacing** (*float[3]*) – A vector containing the spacings of the voxels in i, j and k directions.
- **_subDimensions** (*int[3]*) – A vector containing the numbers of voxels of the subvolume in i, j and k directions.
- **_subOffset** (*int[3]*) – A vector containing the offset (number of voxels) of the first voxel of the subvolume from the first voxel of the original image.
- **_matrix** (*nd.array*) – A matrix representing the origin and the orientation of the image. The matrix is set to identity by default
- **_endian** (*int*) – A variable for the Endian of the scalar values in the image.
- **_numComponents** (*int*) – A variable for the number of components
- **_scalarType** (*int*) – A variable for the scalar type of the voxels
- **_coordinate** (*int*) – A variable for the used coordinate system

**getCoordinateSystem**()
Gets the coordinate system (COORDINATE_RAS or COORDINATE_LPS)

**Returns** The coordinate system (COORDINATE_RAS or COORDINATE_LPS)

**getData**()
Gets the image raw data

**Returns** The image raw data

**getDimensions**()
Gets image dimensions as a tuple of the numbers of pixels in i, j and k directions.

**Returns** number of pixels in i, j, k directions

**getEndian**(*endian*)
Gets the Endianess of the image scalars. (default is ENDIAN_BIG)

**Returns** The endianess of the image scalars

**getImageSize**()

**Gets the size (length) of the byte array for the image data. The size is defined by**
dimensions[0]*dimensions[1]*dimensions[2]*scalarSize*numComponents.

**Returns** The size of the byte array for the image data

**getMatrix**()
Gets the orientation and origin matrix.

**Returns** The 4x4 matrix representing the origin and the orientation of the image.

**getNormals**()
Gets the orientation of the image as an array of the normal vectors for the i, j and k indexes.

**Returns** The image orientation vector concatenated in a matrix per columns

**getNumComponents**()
Gets the number of components for each voxel.

**Returns** the number of components for each voxel.

**getOrigin**()
Gets the coordinates of the origin using an array of positions along the first (R or L), second (A or P) and the third (S) axes.

**Returns** list of origin coordinates

**getScalarSize**()
Gets the size of the scalar type used in the current image data.

**Returns** The size of the scalar type used in the current image data.

**getScalarType**()
Gets the image scalar type.

**Returns** the image scalar type

**getSpacing**()
Gets spacings as an array or list of spacing values in i, j and k directions.

**Returns** list of spacing values

**getSubVolume**()
Gets sub-volume dimensions and offset expressed in pixels in i, j, k directions

**Returns** dims[3], off[3] - 2 lists containing dimensions and offsets of the subvolume

**getSubVolumeSize**()

**Gets the size (length) of the byte array for the subvolume image data. The size is defined by** subDimensions[0]*subDimensions[1]*subDimensions[2]*scalarSize*numComponents.

**Returns** The size (length) of the byte array for the subvolume image data.

**setCoordinateSystem**(*coordSys*)
Sets the coordinate system (COORDINATE_RAS or COORDINATE_LPS)

**Parameters** **coordSys** – The coordinate system (COORDINATE_RAS or COORDINATE_LPS)

**setData**(*rawImgData*)
Sets the image raw data

**Parameters** **rawImgData** – image raw data

**Returns** True if the data were correctly set, False otherwise

**setDimensions**(*dimensions*)
Sets image dimensions by an array of the numbers of pixels in i, j and k directions. SetDimensions() should be called prior to `SetSubVolume()`, since `SetDimensions()` sets subvolume parameters automatically assuming that subvolume = entire volume.

**Parameters** **dimensions** – element list, tuple or array - [rows, cols, channels]

**setEndian**(*endian*)
Sets the Endianess of the image scalars. (default is ENDIAN_BIG)

**Parameters** **endian** – Endianess of the image scalars

**setMatrix**(*matrix*)
Sets the orientation and origin matrix.

**Parameters** **matrix** – a 4x4 matrix representing the origin and the orientation of the image.

**setNormals**(*m*)
Sets the orientation of the image by an array of the normal vectors for the i, j and k indexes.

**Parameters** **m** – a 3x3 matrix containing the normal vectors stored in columns

**setNumComponents**(*num*)
Sets the number of components for each voxel.

**Parameters** **num** – number of components for each voxel

**setOrigin**(*origin*)
Sets the coordinates of the origin by an array of positions along the first (R or L), second (A or P) and the third (S) axes.

**Param** list or array with the origin coordinates

**setScalarType**(*stype*)
Sets the image scalar type. (to one of PixelType)

**Parameters** **stype** – image scalar type

**setScalarTypeToInt16**()
Sets the image scalar type to 16-bit integer.

**setScalarTypeToInt32**()
Sets the image scalar type to 32-bit integer.

**setScalarTypeToInt8**()
Sets the image scalar type to 8-bit integer.

**setScalarTypeToUint16**()
Sets the image scalar type to 16-bit unsigned integer.

**setScalarTypeToUint32**()
Sets the image scalar type to 32-bit unsigned integer.

**setScalarTypeToUint8**()
Sets the image scalar type to 8-bit unsigned integer.

**setSpacing**(*spacing*)
Sets spacings by an array or list of spacing values in i, j and k directions.

**Parameters** **spacing** – array or list of spacing values

**setSubVolume**(*dims*, *off*)
Sets sub-volume dimensions and offset by arrays of the dimensions and the offset. `SetDimensions()`, since `SetDimensions()` reset the subvolume parameters automatically.

**Parameters**

- **dims** – 3-element list or array with the number of subvolume pixels in i, j, k directions
- **off** – 3-element list or array with the subvolume pixels offsets in i, j, k directions

**Returns** True if the subvolume is successfully specified, False if an invalid subvolume is specified.

**class** pygtlink.**SensorMessage**
Bases: `pygtlink.igtl_message_base.MessageBase`

**Variables**

- **_larray** (*int*) – The sensor array len (uint8)
- **_status** (*int*) – The status (uint8)
- **_unit** (*int*) – The unit (uint64)
- **_data** (*list*) – The sensor data (float64[Larray])

**getData**()
Gets sensor data

**Returns** The sensor data

**getLength**()
Gets sensor data length (num elements)

**Returns** The sensor data length

**getStatus**()
Gets sensor status

**Returns** The sensor status

**getUnit**()
Gets sensor data unit

**Returns** The sensor data unit

**setData**(*data*)
Sets sensor data

**Parameters** **data** – The list or array of sensor data

**setLength**(*length*)
Sets sensor data length (num elements)

**Parameters** **length** – The array or list of spacing values

**setStatus**(*status*)
Sets status

**Parameters** **status** – The status to be set

**setUnit**(*unit*)
Sets unit

**Parameters** **unit** – The sensor data unit to be set (must be an int)

**class** pygtlink.**StatusMessage**
Bases: pygtlink.igtl_message_base.MessageBase

**Variables**

- **_code** (*int*) – The command code (uint16)
- **_subCode** (*int*) – The command sub code (int64)
- **_errorName** (*str*) – The error name (char[20])
- **_message** (*str*) – The error message (char[BODY_SIZE - 30])

**getCode**()
Gets the status code

**Returns** The status code

**getErrorName**()
Gets the status error name

**Returns** The status error name

**getMessage**()
Gets the status message

**Returns** The status message

**getSubCode**()
Gets the status subcode

**Returns** The status subcode

**setCode**(*code*)
Sets the status code

**Parameters** **code** – The status code to be set

**setErrorName**(*errorName*)
Sets the status error name

**Parameters** **errorName** – The status error name to be set

**setMessage**(*message*)
Sets the status message

**Parameters** **message** – The status message to be set

**setSubCode**(*subCode*)
Sets the status subcode

**Parameters** **subCode** – The status subcode

**class** pygtlink.**PositionMessage**
Bases: pygtlink.igtl_message_base.MessageBase

The class implements the openIgtLink position message

**Variables**

- **_x** (*float*) – The x component of the position
- **_y** (*float*) – The y component of the position
- **_z** (*float*) – The z component of the position
- **_ox** (*float*) – The first component of the orientation quaternion
- **_oy** (*float*) – The second component of the orientation quaternion
- **_oz** (*float*) – The third component of the orientation quaternion
- **_w** (*float*) – The fourth component of the orientation quaternion

**getPosition**()
Gets the position

**Returns** The position as a list [x, y, z]

**getQuaternion**()
Sets the quaternion

**Returns** The quaternion as a list [ox, oy, oz, w]

**setPosition**(*pos*)
Sets the position

**Parameters** **pos** – The position to be set. It can be a 3-element list or nd.array

**setQuaternion**(*quat*)
Sets the quaternion

**Parameters** **quat** – The quaternion to be set. It can be a 4-element list or nd.array

**class** pygtlink.**SocketServer**
Bases: object

Implementation of IGTL Server

**Variables**

- **_serverSocket** (*socket.socket*) – The server TCP socket listening for incoming connection
- **_clientSocket** (*socket.socket*) – The TCP socket the server opens with the client when a connection request is received over the _serverSocket
- **_serverAddress** (*str*) – The server address
- **_serverPort** (*int*) – The server port

**kill**()
Shut down the socket connection with the client and closes the server socket

**receive**(*length*)
Receives a message of <length> bytes from the IGTL client

**Parameters** **length** (*int*) – The length of the message to be received

**Returns** The received message (a byte string)

**send**(*data*)
Sends data to the IGTL client

**Parameters** **data** – the message to be sent (as a byte string)

**setAddress**(*address*, *port*)
Sets the Server address and port

**Parameters**

- **address** (*str*) – The server address to be set
- **port** (*int*) – The server port to be set

**setTimeout**(*timeout=0*)
Sets receive timeout

**Parameters** **timeout** (*int*) – The receive timeout to be set

**start**()
Creates the server socket and binds it with the server address set with *setAddress()*

**waitForConnection**()
Waits for a connection request to be sent from the client

**class** pygtlink.**ClientSocket**
Bases: object

Implementation of IGTL client

**Variables** **_clientSocket** (*socket.socket*) – The client TCP socket

**connectToServer**(*serverAddress*, *port*)
Connects to the IGTL server

**Parameters**

- **serverAddress** (*str*) – Server Address
- **port** (*int*) – Server Port

**kill**()
Shut down the connection and closes the socket

**receive**(*length*)
Receives a message of <length> bytes from the IGTL server

**Parameters** **length** (*int*) – The length of the message to be received

**Returns** The received message (a byte string)

**send**(*data*)
Sends data to the IGTL server

**Parameters** **data** – the message to be sent (as a byte string)

**setTimeout**(*timeout=0*)
Sets receive timeout

**Parameters** **timeout** (*int*) – The receive timeout to be set

# Python Module Index

## p

# Index

## R



## S



## U



## W